**Lexical Scope Chaining:**

**code example:**

function b() {

console.log(myVar);

}

function a() {

var myVar = 2;

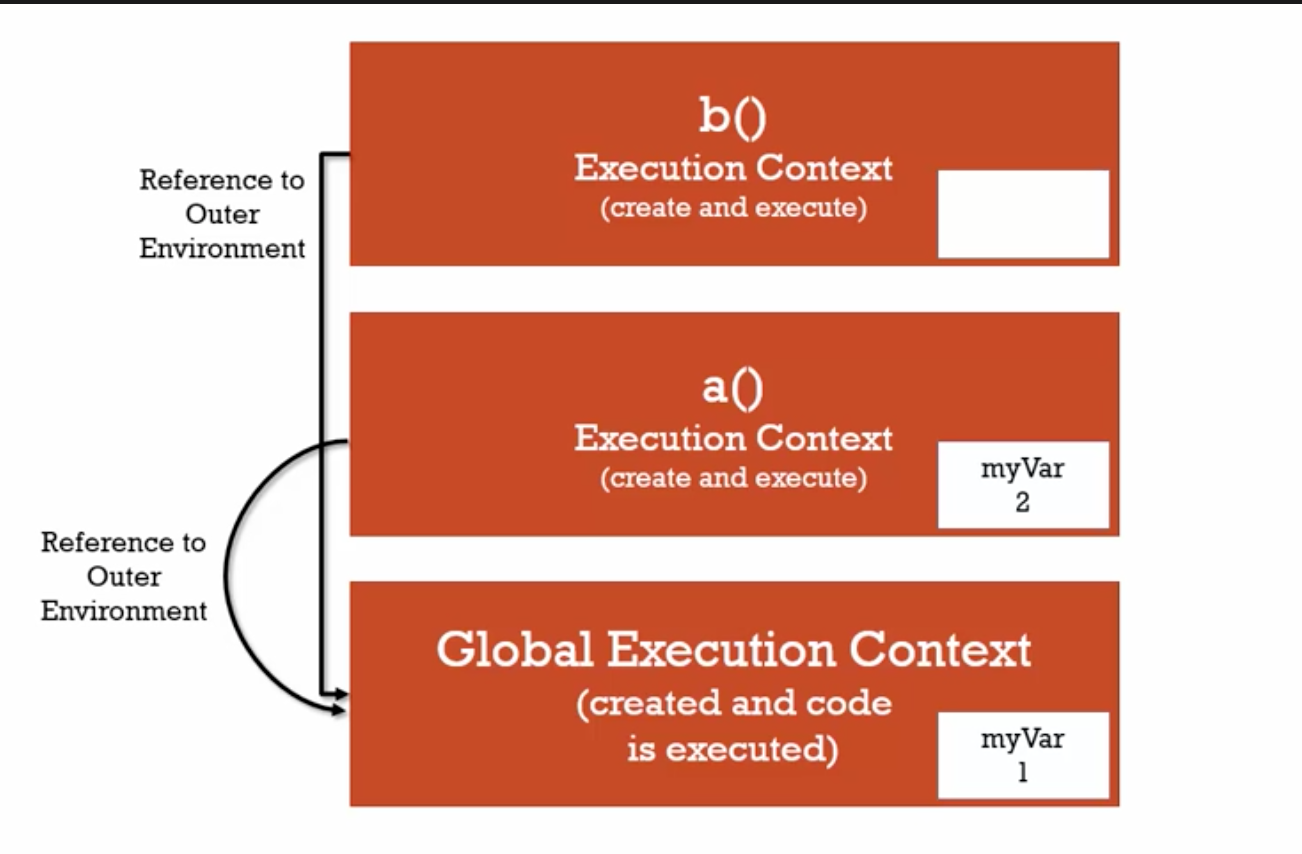
b();

}

var myVar = 1;

a();

**THE SCOPE CHAIN:**



When the global execution will run, it will keep all the functions stored in its memory space but until those functions are invoked they will just sit in the memory space.

Then function a() will be invoked, it will set the value to “2” then it will invoke function b(), which will look for myVar and when it will not find in it’s scope then it will look for myVar in the global execution context. Which in the above example is “myVar  = 1” thus it will display the result of console.log(myVar) = 1. Because of the value is assigned in the global execution context.

Order of execution stack based on the above example:

1. Global Execution Context
2. Function A Stack (Remains open)
3. Function B Stack (Executes & Close) then (Function A stack will close).
4. Global Execution Context will end.